ATTORNEY DOCKET NO. 15104.0001U2
APPLICATION NO. 09/820,185

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of	)	
	)	
Zygmont et al.	)	Group Art Unit: **2122**
	)	
Application No. **09/820,185**	)	Examiner: **Fowlkes, Andre R.**
	)	
Filed: **March 28, 2001**	)	Confirmation No. **3521**
	)	
For: **"SYSTEM AND METHOD FOR METAPROGRAMMING SOFTWARE DEVELOPMENT ENVIRONMENT"**	)	

MAIL STOP FEE AMENDMENT
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

NEEDLE & ROSENBERG, P.C.
Customer No. 23859

DECLARATION OF DAVID A. ZYGMONT UNDER 37 C.F.R. § 1.131

Sir:

I, David A. Zygmont, declare that:

1. I am over the age of twenty-one (21) years. I am not suffering from any disabilities, I am competent to make this Declaration, and I have personal knowledge of the facts set forth herein.

2. I am a named co-inventor in the above-identified application ("Application") and am a co-inventor of the subject matter described and claimed therein.

3. Prior to the filing date of U.S. Patent No. 6,715,145 to Bowman-Amuah ("Bowman") (August 31, 1999), which is also the priority date of Bowman, I had conceived and reduced to practice the invention described and claimed in the Application in the United States at least as early as September 23, 1998, as evidenced by the following:

I conceived and reduced the invention to practice as shown in Exhibits 1, 2, 3, 4, 5, 6, and 7, attached.

Exhibit 1 shows a computer file created on September 23, 1998, named Class_p.cpp, which illustrates a metaprogram to be combined with an object model by a meta-machine to generate a software system as in the Application. Metaprograms in the application are used to define a computer system architecture and are composed of a traditional programming language as well as metacode as described in the Application. In Exhibit 1, the traditional programming language is C++, and the metacode is delimited by the $ symbol.

Exhibit 2 shows a computer file named class_p.cpp-history.txt, which is the Visual Source Safe (VSS) history for Class_p.cpp. Exhibit 2 shows the Class_p.cpp file being created on September 23, 1998.

Exhibit 3 shows a computer file created on January 13, 1999, named SQL7x_spl.cmt, which illustrates a metaprogram to be combined with an object model by a meta-machine to generate a software system as in the Application.

Exhibit 4 shows a computer file named sql7x_sql.cmt-history.txt, which is a VSS file history showing that the file named SQL7x_spl.cmt was created on January 13, 1999.

Exhibit 5 shows a Class Diagram representing the model of one embodiment of the invention of the Application. Exhibit 5 was created on October 16, 1998.

Exhibit 6 shows a Class Diagram representing the model generator of one embodiment of the invention of the Application. Exhibit 6 was created during the month of October, 1998.

Exhibit 7 is a Use Case Diagram representing one embodiment of the invention of the Application. Exhibit 7 was created during the month of October, 1998.

Exhibit 8 is a Class Diagram of the code generator of one embodiment of the

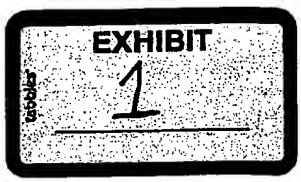
invention of the Application. Exhibit 8 was created during the month of October, 1998.

4. I further declare that all statements made herein of my own knowledge are true, and that all statements made on information and belief are believed to be true; and, further, that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under § 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or document or any patent issuing therefrom.

6/3/2005
DATE

David A. Zygmont

```
/* ------------------------------------------------------------------------
 * File Name : $CLASS_NAME$_p.cpp
 * Platform:
 *  -- Windows NT 4.0 | Windows 95
 * Dialect:
 *  -- MS VC++ 5.0
 * Template: ATL
 * API set:  MS Win32, ADO
 * Dev Lib:  MFC lib
 * Build Info:
 *   -- other required files(header, resource, etc)
 *------------------------------------------------------------------------
 */

/*
 * $CLASS_NAME$: persistent business object class
 * $CLASS_DESCRIPTION$
 * This file is generated
 * Based on version $$VERSION$$ of the $$PROJECT_CODE$$ OO model
 * --- DO NOT MODIFY ---
 *
 * @version $$VERSION$$
 * @author  n/a
 */

#include "stdafx.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

// include persistent object system
#include "POSystem.h"

// #include class delaration
#include "$CLASS_NAME$_p.h"
$BEG_IF_SUBCLASS$
#include "$SUPER_CLASS_NAME$_p.h"
$END_IF_SUBCLASS$
// 1 -> 1 or N -> 1
$BEG_FOR_EACH_ASSOC_OR_AGGR_TO_1$
$BEG_IF_RELATION_RELATED_CLASS_PERSISTENT$
#include "$RELATED_CLASS_NAME$_p.h"
$END_IF_RELATION_RELATED_CLASS_PERSISTENT$
$END_FOR_EACH_ASSOC_OR_AGGR_TO_1$
// 1 -> N
$BEG_FOR_EACH_ASSOC_OR_AGGR_1_TO_N$
$BEG_IF_RELATION_RELATED_CLASS_PERSISTENT$
#include "$RELATED_CLASS_NAME$_p.h"
#include "$RELATED_CLASS_NAME$_pc.h"
$END_IF_RELATION_RELATED_CLASS_PERSISTENT$
$END_FOR_EACH_ASSOC_OR_AGGR_1_TO_N$
// N -> N
$BEG_FOR_EACH_ASSOC_OR_AGGR_N_TO_N$
$BEG_IF_RELATION_RELATED_CLASS_PERSISTENT$
#include "$RELATED_CLASS_NAME$_p.h"
#include "$RELATED_CLASS_NAME$_pc.h"
$END_IF_RELATION_RELATED_CLASS_PERSISTEN$
$END_FOR_EACH_ASSOC_OR_AGGR_N_TO_N$
```

```
// static variables
static LPCTSTR _SelectText =
"SELECT $CLASS_NAME$.$CLASS_PK_NAME$ \
   $BEG_FOR_EACH_ASSOC_OR_AGGR_TO_1$,
$CLASS_NAME$.$RELATED_CLASS_ROLE_FK_NAME$$END_FOR_EACH_ASSOC_OR_AGGR_TO_1$ \
   $BEG_FOR_EACH_PERSISTENT_ATTR$,
$CLASS_NAME$.$ATTR_NAME$$END_FOR_EACH_PERSISTENT_ATTR$ \
  FROM $CLASS_NAME$ ";

static LPCTSTR _InsertText =
"INSERT INTO $CLASS_NAME$ \
    ($CLASS_PK_NAME$ \
     $BEG_FOR_EACH_ASSOC_OR_AGGR_TO_1$,
$RELATED_CLASS_ROLE_FK_NAME$$END_FOR_EACH_ASSOC_OR_AGGR_TO_1$ \
     $BEG_FOR_EACH_PERSISTENT_ATTR$, $ATTR_NAME$$END_FOR_EACH_PERSISTENT_ATTR$) ";

static LPCTSTR _UpdateText =
"UPDATE $CLASS_NAME$ SET ";

static LPCTSTR _DeleteText =
"DELETE FROM $CLASS_NAME$ WHERE $CLASS_PK_NAME$ = ";


// create NULL object
$CLASS_NAME$_p*
$CLASS_NAME$_p::CreateNullObj()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    $CLASS_NAME$_p* pNullPO = new $CLASS_NAME$_p();
    pNullPO->m_isNULL = TRUE;
    pNullPO->SetNew(FALSE);
    pNullPO->SetModified(FALSE);
    pNullPO->m_isPopulated = TRUE;

    return pNullPO;
}

// check nullness
BOOL
$CLASS_NAME$_p::IsNull() const
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    return (m_isNULL);
}

// Constructors, Destructor and Assignment Operator
$CLASS_NAME$_p::$CLASS_NAME$_p() :
    PObject(),
    m_pos(POSystem::GetInstance())
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    this->SetNew(TRUE);
    this->SetModified(TRUE);
    m_isNULL = FALSE;
    m_isPopulated = FALSE;
$BEG_IFNOT_SUBCLASS$
    m_$CLASS_PK_NAME$ = PObject::GeneratePID();
$END_IFNOT_SUBCLASS$
}

$CLASS_NAME$_p::$CLASS_NAME$_p(LPCTSTR pid) :
    PObject(),
    m_pos(POSystem::GetInstance())
```

```
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    m_sqlSelectStmt = _SelectText;
    m_sqlSelectStmt += " WHERE $CLASS_NAME$.$CLASS_PK_NAME$ = '";
    m_sqlSelectStmt += pid;
    m_sqlSelectStmt += "'";
    m_$CLASS_PK_NAME$ = pid;

    this->SetModified(FALSE);
    this->SetNew(FALSE);
    if(_tcslen(pid) > 0)
        m_isNULL = FALSE;
    else
        m_isNULL = TRUE;
    m_isPopulated = FALSE;
}

void $CLASS_NAME$_p::PopulateSelf() const
{
    if(m_isPopulated) return;

    if(m_sqlSelectStmt.GetLength() == 0) return;

    ADODB::_RecordsetPtr pRecordset
        = m_pos.Execute(m_sqlSelectStmt);
    $CLASS_NAME$_p* pMutableObj
        = ($CLASS_NAME$_p*)this;
    if(!pRecordset->GetadoEOF())
    {
        pMutableObj->Populate(pRecordset);
    }
    else
    {
        pMutableObj->SetNew(FALSE);
        pMutableObj->SetModified(FALSE);
        pMutableObj->m_isNULL = FALSE;
    }

    pMutableObj->m_isPopulated = TRUE;
}

$CLASS_NAME$_p::$CLASS_NAME$_p(ADODB::_RecordsetPtr& pRecordset) :
    PObject(),
    m_pos(POSystem::GetInstance())
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    if(!pRecordset->GetadoEOF())
    {
        this->Populate(pRecordset);
        this->SetModified(FALSE);
        this->SetNew(FALSE);
        m_isNULL = FALSE;
    }
    else
    {
        this->SetModified(FALSE);
        this->SetNew(FALSE);
        m_isNULL = TRUE;
    }
    m_isPopulated = TRUE;
}
```

```
// ctor by super class
$BEG_IF_SUBCLASS$
$CLASS_NAME$_p::$CLASS_NAME$_p(const $SUPER_CLASS_NAME$_p& superPO) :
    PObject(),
    m_pos(POSystem::GetInstance())
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    m_sqlSelectStmt = _SelectText;
    m_sqlSelectStmt += " WHERE $CLASS_NAME$.$CLASS_PK_NAME$ = '";
    m_sqlSelectStmt += superPO.GetPID();
    m_sqlSelectStmt += "'";

    this->SetModified(FALSE);
    this->SetNew(FALSE);
    m_isNULL = FALSE;
    m_isPopulated = FALSE;
}
$END_IF_SUBCLASS$

// dtor
$CLASS_NAME$_p::~$CLASS_NAME$_p()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
}

void $CLASS_NAME$_p::SetPID(LPCTSTR pid)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    m_$CLASS_PK_NAME$ = pid;
}

CString $CLASS_NAME$_p::GetPID() const
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    return m_$CLASS_PK_NAME$;
}

BOOL $CLASS_NAME$_p::Populate(ADODB::_RecordsetPtr& pRecordset)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    ADODB::FieldsPtr pFields = pRecordset->GetFields();

    _variant_t varItem;
    varItem
        = pFields->GetItem(_variant_t("$CLASS_PK_NAME$"))->GetValue();
    m_$CLASS_PK_NAME$ = POSystem::VARIANTToKeyType(varItem);


    // do not assign any value if the DB value is NULL
$BEG_FOR_EACH_PERSISTENT_ATTR$
    varItem = pFields->GetItem(_variant_t("$ATTR_NAME$"))->GetValue();
    m_$ATTR_NAME$ = POSystem::VARIANTTo$ATTR_TYPE$(varItem);
$END_FOR_EACH_PERSISTENT_ATTR$
$BEG_FOR_EACH_ASSOC_OR_AGGR_TO_1$
    varItem =
pFields->GetItem(_variant_t("$RELATED_CLASS_ROLE_FK_NAME$"))->GetValue();
    m_$RELATED_CLASS_ROLE_FK_NAME$ = POSystem::VARIANTToKeyType(varItem);

$END_FOR_EACH_ASSOC_OR_AGGR_TO_1$

    this->SetModified(FALSE);
    this->SetNew(FALSE);
```

```
    m_isNULL = FALSE;
    m_isPopulated = TRUE;

    return TRUE;
}

// basic persistent operation
BOOL
$CLASS_NAME$_p::Save()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    // check if the save operation need to be performed
    if(!this->IsModified())
        return TRUE;

    if(this->IsNew())
    {
        // insert
        CString valueStr(" VALUES(");
        valueStr += "'";
        valueStr += m_$CLASS_PK_NAME$;
        valueStr += "' ";
$BEG_FOR_EACH_ASSOC_OR_AGGR_TO_1$
        valueStr += ", ";
        if(m_$RELATED_CLASS_ROLE_FK_NAME$.GetLength())
        {
            valueStr += "'";
            valueStr += m_$RELATED_CLASS_ROLE_FK_NAME$;
            valueStr += "'";
        }
        else
        {
            valueStr += "NULL";
        }
$END_FOR_EACH_ASSOC_OR_AGGR_TO_1$
$BEG_FOR_EACH_PERSISTENT_ATTR$
        valueStr += ", ";
        valueStr += POSystem::$ATTR_TYPE_1ST_CAP$ToString(m_$ATTR_NAME$);
$END_FOR_EACH_PERSISTENT_ATTR$
        valueStr += ")";
        m_pos.Execute(_InsertText + valueStr);
    }
    else
    {
        // update
        CString setStr;
$BEG_FOR_EACH_PERSISTENT_ATTR$
        setStr +=
"$BEG_IFNOT_LOOP_ITEM_FIRST$,$END_IFNOT_LOOP_ITEM_FIRST$$ATTR_NAME$=";
        setStr += POSystem::$ATTR_TYPE_1ST_CAP$ToString(m_$ATTR_NAME$);
$END_FOR_EACH_PERSISTENT_ATTR$
$BEG_FOR_EACH_ASSOC_OR_AGGR_TO_1$
        setStr += ", $RELATED_CLASS_ROLE_FK_NAME$=";
        if(m_$RELATED_CLASS_ROLE_FK_NAME$.GetLength())
        {
            setStr += "'";
            setStr += m_$RELATED_CLASS_ROLE_FK_NAME$;
            setStr += "'";
        }
        else
        {
            setStr += "NULL";
        }
```

```
$END_FOR_EACH_ASSOC_OR_AGGR_TO_1$
        setStr += " WHERE $CLASS_PK_NAME$ = '";
        setStr += m_$CLASS_PK_NAME$;
        setStr += "'";

        m_pos.Execute(_UpdateText + setStr);
    }

    this->SetModified(FALSE);
    this->SetNew(FALSE);

    return TRUE;
}

// reload from data storage
BOOL
$CLASS_NAME$_p::Reload()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    CString sqlText(_SelectText);
    sqlText += " WHERE $CLASS_NAME$.$CLASS_PK_NAME$ = '";
    sqlText += this->GetPID();
    sqlText += "'";

    ADODB::_RecordsetPtr pRecordset
        = m_pos.Execute(sqlText);
    if(!pRecordset->GetadoEOF())
    {
        this->Populate(pRecordset);
        this->SetModified(FALSE);
        this->SetNew(FALSE);
        m_isNULL = FALSE;
    }
    else
    {
        this->SetModified(TRUE);
        this->SetNew(TRUE);
        m_isNULL = TRUE;
    }
    return TRUE;
}

// delete from data storage
BOOL
$CLASS_NAME$_p::Delete()
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    if(!m_isPopulated) this->PopulateSelf();

    // m_pos.BeginTrans();
    // reset all N -> N relations
$BEG_FOR_EACH_ASSOC_OR_AGGR_N_TO_N$
$BEG_IF_RELATION_RELATED_CLASS_PERSISTENT$
    {
        // delete the relation
        CString deleteStmt("DELETE FROM $RELATION_NAME$ WHERE
$RELATION_NAME$.$CLASS_ROLE_FK_NAME$ = '");
        deleteStmt += this->GetPID();
        deleteStmt += "'";
        ADODB::_RecordsetPtr pRecordset
            = m_pos.Execute(deleteStmt);
    }
```

```
$END_IF_RELATION_RELATED_CLASS_PERSISTENT$
$END_FOR_EACH_ASSOC_OR_AGGR_N_TO_N$

    // reset all 1 -> N association relations
$BEG_FOR_EACH_ASSOCIATION_1_TO_N$
$BEG_IF_RELATION_RELATED_CLASS_PERSISTENT$
    {
        // reset relation for all the related classes
        CString updateStmt("UPDATE $RELATED_CLASS_NAME$ SET $CLASS_ROLE_FK_NAME$ =
NULL WHERE $CLASS_ROLE_FK_NAME$ = '");
        updateStmt += this->GetPID();
        updateStmt += "'";
        ADODB::_RecordsetPtr pRecordset = m_pos.Execute(updateStmt);
    }
$END_IF_RELATION_RELATED_CLASS_PERSISTENT$
$END_FOR_EACH_ASSOCIATION_1_TO_N$

    // delete all 1 -> N aggregated objects
$BEG_FOR_EACH_AGGREGATION_1_TO_N$
$BEG_IF_RELATION_THIS_CLASS_AGGREGATE$
$BEG_IF_RELATION_RELATED_CLASS_PERSISTENT$
    {
        $RELATED_CLASS_NAME$_pc* pPOList =
this->Get$RELATED_CLASS_ROLE_NAME_1ST_CAP$List();
        pPOList->DeleteAll();
        delete pPOList;
    }
$END_IF_RELATION_RELATED_CLASS_PERSISTENT$
$END_IF_RELATION_THIS_CLASS_AGGREGATE$
$END_FOR_EACH_AGGREGATION_1_TO_N$

    // delete all N -> 1 or 1 -> 1 aggregated objects
$BEG_FOR_EACH_AGGREGATION_TO_1$
$BEG_IF_RELATION_THIS_CLASS_AGGREGATE$
$BEG_IF_RELATION_RELATED_CLASS_PERSISTENT$
    {
        if(this->Has$RELATED_CLASS_ROLE_NAME_1ST_CAP$())
        {
            $RELATED_CLASS_NAME$_p* pPO =
this->Get$RELATED_CLASS_ROLE_NAME_1ST_CAP$();
            pPO->Delete();
            delete pPO;
        }
    }
$END_IF_RELATION_RELATED_CLASS_PERSISTENT$
$END_IF_RELATION_THIS_CLASS_AGGREGATE$
$END_FOR_EACH_AGGREGATION_TO_1$
$BEG_IF_SUBCLASS$
    // delete super class
    {
        CString delStmt = "DELETE FROM $SUPER_CLASS_NAME$ WHERE $CLASS_PK_NAME$ =
'";
        delStmt += this->GetPID();
        delStmt += "'";
        m_pos.Execute(delStmt);
    }
$END_IF_SUBCLASS$
    // delete self
    CString idStr = "'" + m_$CLASS_PK_NAME$ + "'";
    m_pos.Execute(_DeleteText + idStr);

    // m_pos.CommitTrans();
```

```
    SetModified(TRUE);
    SetNew(TRUE);

    return TRUE;
}

// copy from other object
void
$CLASS_NAME$_p::CopyAttrFrom(const $CLASS_NAME$_p& po)
{
    // local attributes
$BEG_FOR_EACH_PERSISTENT_ATTR$
    this->Set$ATTR_NAME_1ST_CAP$(po.Get$ATTR_NAME_1ST_CAP$());
$END_FOR_EACH_PERSISTENT_ATTR$
}

$BEG_IF_SUBCLASS$
// treat inheritance as 1 -> 1 relation
$SUPER_CLASS_NAME$_p* $CLASS_NAME$_p::Get$SUPER_CLASS_NAME$() const
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    if(!m_isPopulated) this->PopulateSelf();

    ASSERT(m_$CLASS_PK_NAME$.GetLength() > 0);

    return new $SUPER_CLASS_NAME$_p(m_$CLASS_PK_NAME$);
}

// set up relation
void $CLASS_NAME$_p::Set$SUPER_CLASS_NAME$(const $SUPER_CLASS_NAME$_p& po)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    if(!m_isPopulated) this->PopulateSelf();
    this->m_$CLASS_PK_NAME$ = po.GetPID();
    this->SetModified(TRUE);
}
$END_IF_SUBCLASS$

// Traversal 1 -> N (ASSOC_OR_AGGR)
$BEG_FOR_EACH_ASSOC_OR_AGGR_1_TO_N$
$BEG_IF_RELATION_RELATED_CLASS_PERSISTENT$
$BEG_IF_RELATION_RELATED_CLASS_NAVIGABLE$
$RELATED_CLASS_NAME$_pc*
$CLASS_NAME$_p::Get$RELATED_CLASS_ROLE_NAME_1ST_CAP$List() const
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    if(!m_isPopulated) this->PopulateSelf();
    CString boolExpression("$RELATED_CLASS_NAME$.$CLASS_ROLE_FK_NAME$ = ");
    boolExpression += "'";
    boolExpression += this->GetPID();
    boolExpression += "'";

    $RELATED_CLASS_NAME$_pc* pPOList = new $RELATED_CLASS_NAME$_pc();
    pPOList->Search(boolExpression);
    return pPOList;
}

// Set up relationship 1 -> N
void
$CLASS_NAME$_p::Set$RELATED_CLASS_ROLE_NAME_1ST_CAP$($RELATED_CLASS_NAME$_p& po)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
```

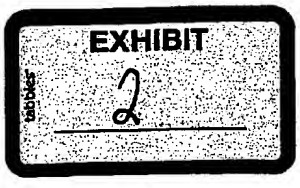
```
    if(!m_isPopulated) this->PopulateSelf();
    if(po.IsNull())
    {
        // reset relation for all the related classes
        CString updateStmt("UPDATE $RELATED_CLASS_NAME$ SET $CLASS_ROLE_FK_NAME$ =
NULL WHERE $CLASS_ROLE_FK_NAME$ = '");
        updateStmt += this->GetPID();
        updateStmt += "'";
        ADODB::_RecordsetPtr pRecordset = m_pos.Execute(updateStmt);
    }
    else
    {
        if(!po.m_isPopulated) po.PopulateSelf();
        po.m_$CLASS_ROLE_FK_NAME$ = this->GetPID();
        po.SetModified(TRUE);
        po.Save();

        //CString updateStmt("UPDATE $RELATED_CLASS_NAME$ SET $CLASS_ROLE_FK_NAME$ =
'");
        //updateStmt += this->GetPID();
        //updateStmt += "' WHERE $RELATED_CLASS_PK_NAME$ = '";
        //updateStmt += po.GetPID();
        //updateStmt += "'";
        //ADODB::_RecordsetPtr pRecordset = m_pos.Execute(updateStmt);

    }
}
$END_IF_RELATION_RELATED_CLASS_NAVIGABLE$
$END_IF_RELATION_RELATED_CLASS_PERSISTENT$
$END_FOR_EACH_ASSOC_OR_AGGR_1_TO_N$

// Traversal N -> N (ASSOC_OR_AGGR)
$BEG_FOR_EACH_ASSOC_OR_AGGR_N_TO_N$
$BEG_IF_RELATION_RELATED_CLASS_PERSISTENT$
$BEG_IF_RELATION_RELATED_CLASS_NAVIGABLE$
$RELATED_CLASS_NAME$_pc*
$CLASS_NAME$_p::Get$RELATED_CLASS_ROLE_NAME_1ST_CAP$List() const
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    if(!m_isPopulated) this->PopulateSelf();
    CString boolExpression("$RELATION_NAME$.$CLASS_ROLE_FK_NAME$ = '");
    boolExpression += this->GetPID();
    boolExpression += "' AND ";
    boolExpression += " $RELATION_NAME$.$RELATED_CLASS_ROLE_FK_NAME$ =
$RELATED_CLASS_NAME$.$RELATED_CLASS_PK_NAME$";

    CString addTabNames("$RELATION_NAME$");

    $RELATED_CLASS_NAME$_pc* pPOList = new $RELATED_CLASS_NAME$_pc();
    pPOList->Search(boolExpression, addTabNames);
    return pPOList;
}

// Set up relationship with ONE N -> N related object
void
$CLASS_NAME$_p::Set$RELATED_CLASS_ROLE_NAME_1ST_CAP$(const $RELATED_CLASS_NAME$_p&
po)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    if(!m_isPopulated) this->PopulateSelf();
    if(po.IsNull())
```

```
    {
        // delete the relation
        CString deleteStmt("DELETE FROM $RELATION_NAME$ WHERE
$RELATION_NAME$.$CLASS_ROLE_FK_NAME$ = '");
        deleteStmt += this->GetPID();
        deleteStmt += "'";
        ADODB::_RecordsetPtr pRecordset
            = m_pos.Execute(deleteStmt);
        return;
    }

    CString selectStmt("SELECT * FROM $RELATION_NAME$ WHERE
$RELATION_NAME$.$CLASS_ROLE_FK_NAME$ = '");
    selectStmt += this->GetPID();
    selectStmt += "' AND $RELATION_NAME$.$RELATED_CLASS_ROLE_FK_NAME$ = '";
    selectStmt += po.GetPID();
    selectStmt += "'";
    ADODB::_RecordsetPtr pRecordset
        = m_pos.Execute(selectStmt);
    if(pRecordset->GetadoEOF())
    {
        CString insertStmt("INSERT INTO $RELATION_NAME$ ($CLASS_ROLE_FK_NAME$,
$RELATED_CLASS_ROLE_FK_NAME$) VALUES ('");
        insertStmt += this->GetPID();
        insertStmt += "', '";
        insertStmt += po.GetPID();
        insertStmt += "')";
        m_pos.Execute(insertStmt);
    }
}
$END_IF_RELATION_RELATED_CLASS_NAVIGABLE$
$END_IF_RELATION_RELATED_CLASS_PERSISTENT$
$END_FOR_EACH_ASSOC_OR_AGGR_N_TO_N$

// Traversal N -> 1 or 1 -> 1 (ASSOC_OR_AGGR)
$BEG_FOR_EACH_ASSOC_OR_AGGR_TO_1$
$BEG_IF_RELATION_RELATED_CLASS_PERSISTENT$
$BEG_IF_RELATION_RELATED_CLASS_NAVIGABLE$
BOOL
$CLASS_NAME$_p::Has$RELATED_CLASS_ROLE_NAME_1ST_CAP$() const
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    if(!m_isPopulated) this->PopulateSelf();
    return (m_$RELATED_CLASS_ROLE_FK_NAME$.GetLength() != 0);
}

$RELATED_CLASS_NAME$_p*
$CLASS_NAME$_p::Get$RELATED_CLASS_ROLE_NAME_1ST_CAP$() const
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    if(!m_isPopulated) this->PopulateSelf();
    $RELATED_CLASS_NAME$_p* pPO
        = new $RELATED_CLASS_NAME$_p(m_$RELATED_CLASS_ROLE_FK_NAME$);

    if(pPO->IsNull()) {
        delete pPO;
        return NULL;
    }
    return pPO;
}
```

```
// Set up relationship
void
$CLASS_NAME$_p::Set$RELATED_CLASS_ROLE_NAME_1ST_CAP$(const $RELATED_CLASS_NAME$_p&
po)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    if(!m_isPopulated) this->PopulateSelf();
    m_$RELATED_CLASS_ROLE_FK_NAME$ = po.GetPID();
    this->SetModified(TRUE);
}
$END_IF_RELATION_RELATED_CLASS_NAVIGABLE$
$END_IF_RELATION_RELATED_CLASS_PERSISTENT$
$END_FOR_EACH_ASSOC_OR_AGGR_TO_1$


/**
* Attribute accessors and mutators
* Scope: public
* @param     na
* @return    na
* @exception na
*/
// Attribute Accessors and Mutators
$BEG_FOR_EACH_PERSISTENT_ATTR$
const $ATTR_TYPE$ $CLASS_NAME$_p::Get$ATTR_NAME_1ST_CAP$() const
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    if(!m_isPopulated) this->PopulateSelf();
    return m_$ATTR_NAME$;
}
void
$CLASS_NAME$_p::Set$ATTR_NAME_1ST_CAP$(const $ATTR_TYPE$ attr)
{
    AFX_MANAGE_STATE(AfxGetStaticModuleState());

    if(!m_isPopulated) this->PopulateSelf();
    m_$ATTR_NAME$ = attr;
    this->SetModified(TRUE);
}
$END_FOR_EACH_PERSISTENT_ATTR$
```

$/pRobot/data_access/ADO_VC++5.0/Class_p.cpp

	Wwang	8/08/02	4:25p	Labeled '1.86preview5'
	Wwang	1/28/02	2:56p	Labeled 'v 1.85'
	Wwang	1/28/02	12:45p	Labeled 'pre 1.85 '
	Wwang	1/28/02	12:43p	Labeled 'pre 1.85'
	Wwang	8/01/01	4:12p	Labeled 'BranchOut-4.0'
2	Wwang	11/02/98	10:46a	Branched at version 2
1	Wwang	9/23/98	12:23a	Created Class_p.cpp



```
/* ============================================================
 *   Database name:  <%=mcomponent.name%>
 *   DBMS name:      Oracle 8.x
 *   Created on:     $$DATETIME$$
 * ============================================================
 */
<% var Counter = 1; %>
<% var theMClasses = mcomponent.getMetaClasses();
   for (i = 0; i < theMClasses.Count; i++) {
       var theMClass = theMClasses.Item(i);
       if(theMClass.persistence != xyPERSISTENT) {
           continue;
       }
       var theAttrs = theMClass.getAttributes();
%>
/* ============================================================
 *   Table: <%=theMClass.name%>
 * ============================================================
 */
create table <%=theMClass.name%>
(
    <%=theMClass.name%>Id      varchar(40)       not null,
<%
       for(j = 0; j < theAttrs.Count; j++) {
           var theAttr = theAttrs.Item(j);
           if(theAttr.persistence == xyPERSISTENT) {
               dbType = AppGen.getImpDataType("SQL7",theAttr.dataType);
               dbTypeSize = "";
               if(theAttr.size != "") {
                   dbTypeSize = "(" + theAttr.size + ")";
               }
%>
    <%=theAttr.name%> <%=dbType%><%=dbTypeSize%><%if(!theAttr.isNullable){%> not
null<%}else{%> null<%}%>,
<%
           } // end if(theAttr.persistence == xyPERSISTENT)
       } // for(j = 0; j < theAttrs.Count; j++)

       // find out foreign keys
       var theRoles = theMClass.getRoles();
       for(j = 0; j < theRoles.Count; j++) {
           var theRole = theRoles.Item(j);
           var theRelatedRole = theRole.getRelatedRole();
           if(theRole.getAssociation().hasLinkClass()) {
               // the FK to the link table will be in the current table
               //  iff [*] to [0,1 or 1]
               //  otherwise the FK's will be in the link table
               if(theRelatedRole.multiplicity == xyEXACTLY_ONE
                  || theRelatedRole.multiplicity == xyOPTIONAL) {
%>
    <%=theRole.getAssociation().getLinkClass().name%>Id
varchar(40)<%if(theRelatedRole.multiplicity==xyEXACTLY_ONE){%> not
null<%}else{%> null<%}%>,
<%
               }
               continue;
           }
```

EXHIBIT 3

```
            var hasFK = false;
            if(theRelatedRole.getMetaClass().persistence == xyTRANSIENT) {
                continue;
            }
            if(theRelatedRole.multiplicity == xyEXACTLY_ONE
               || theRelatedRole.multiplicity == xyOPTIONAL) {
               if(theRole.isAggregate) {
                   // FK always favor to aggregating class
                   hasFK = true;
               } else if(theRelatedRole.isAggregate) {
                   if (theRole.multiplicity == xyZERO_OR_MORE ||
theRole.multiplcity == xyONE_OR_MORE || theRole.multiplcity == xyMANY)
                   {
                       hasFK = true;
                   }
               } else if(theRelatedRole.multiplicity == theRole.multiplicity) {
                   // we have 1-1 or 0,1-0,1 association
                   //   we need to pick a side
                   if(theRole.objId ==
theRole.getAssociation().getRoles().Item(0).objId) {
                       // theRole is the first role in the association
                       hasFK = true;
                   }
               } else if(theRelatedRole.multiplicity == xyEXACTLY_ONE) {
                   hasFK = true;
               } else if(theRole.multiplicity != xyEXACTLY_ONE) {
                   hasFK = true;
               }
            }
            if(hasFK) {
%>
    <%=theRelatedRole.roleName%>Id
varchar(40)<%if(theRelatedRole.multiplicity==xyEXACTLY_ONE){%> not
null<%}else{%> null<%}%>,
<%
            } // end if(hasFK)
        } // end for(j = 0; j < theRoles.Count; j++)

        // if the class is a link class
        //   more foreign key(s) need to be added
        if(theMClass.hasLinkAssociation()) {
            var theRole1 = theMClass.getLinkAssociation().getRoles().Item(0);
            var theRole2 = theMClass.getLinkAssociation().getRoles().Item(1);
            if(theRole1.multiplicity == xyZERO_OR_MORE
               || theRole1.multiplicity == xyONE_OR_MORE
               || theRole1.multiplicity == xyMANY)
            {
%>
    <%=theRole2.roleName%>Id varchar(40) not null,
<%
            }//end if(theRole1.multiplicity == xyZERO_OR_MORE...)
            if(theRole2.multiplicity == xyZERO_OR_MORE
               || theRole2.multiplicity == xyONE_OR_MORE
               || theRole2.multiplicity == xyMANY)
            {
%>
    <%=theRole1.roleName%>Id varchar(40) not null,
```

```
<%
            }//end if(theRole2.multiplicity == xyZERO_OR_MORE...)
        } // end if(theMClass.hasLinkAssociation())
%>
    constraint PK_<%=theMClass.name%> primary key (<%=theMClass.name%>Id)
)
go
<%      // for each [many] to [1 or 0.1] associatioin
        // create index
        var theRoles = theMClass.getRoles();
        for(j = 0; j < theRoles.Count; j++) {
            var theRole = theRoles.Item(j);
            var theRelatedRole = theRole.getRelatedRole();
            var hasFK = false;
            if(theRole.getAssociation().hasLinkClass()) {
                // the FK to the link table will be in the current table
                //  iff [*] to [0,1 or 1]
                //  otherwise the FK's will be in the link table
                if(theRelatedRole.multiplicity == xyEXACTLY_ONE
                   || theRelatedRole.multiplicity == xyOPTIONAL) {
%>
/* ========================================================
 *   Index: Rel_<%=theRole.roleName%><%=Counter%><%Counter++%>_FK
 * ========================================================
 */
create index Rel_<%=theRole.roleName%><%=Counter%><%Counter++%>_FK on
<%=theMClass.name%> (<%=theRole.roleName%>Id)
go
<%
                }
                continue;
            }
            if(theRelatedRole.getMetaClass().persistence == xyTRANSIENT) {
                continue;
            }
            if(theRelatedRole.multiplicity == xyEXACTLY_ONE
               || theRelatedRole.multiplicity == xyOPTIONAL) {
               if(theRole.isAggregate) {
                   // FK always favor to aggregating class
                   hasFK = true;
               } else if(theRelatedRole.isAggregate) {
                   if (theRole.multiplicity == xyZERO_OR_MORE
                       || theRole.multiplcity == xyONE_OR_MORE
                       || theRole.multiplcity == xyMANY)
                   {
                       hasFK = true;
                   }
               } else if(theRelatedRole.multiplicity == theRole.multiplicity) {
                   // we have 1-1 or 0,1-0,1 association
                   //   we need to pick a side
                   if(theRole.objId ==
theRole.getAssociation().getRoles().Item(0).objId) {
                       // theRole is the first role in the association
                       hasFK = true;
                   }
               } else if(theRelatedRole.multiplicity == xyEXACTLY_ONE) {
                   hasFK = true;
```

```
            } else if(theRole.multiplicity != xyEXACTLY_ONE) {
                hasFK = true;
            }
        }
        if(hasFK) {
%>
/* ============================================================
 *   Index: Rel_<%=theRole.roleName%><%=Counter%><%Counter++%>_FK
 * ============================================================
 */
create index Rel_<%=theRole.roleName%><%=Counter%><%Counter++%>_FK on
<%=theMClass.name%> (<%=theRelatedRole.roleName%>Id);
<%
        } // end if(hasFK)
    } // end for(j = 0; j < theRoles.Count; j++)
    // if the class is a link class
    //   more foreign key index need to be added
    if(theMClass.hasLinkAssociation()) {
        var theRole1 = theMClass.getLinkAssociation().getRoles().Item(0);
        var theRole2 = theMClass.getLinkAssociation().getRoles().Item(1);
        if(theRole1.multiplicity == xyZERO_OR_MORE
           || theRole1.multiplicity == xyONE_OR_MORE
           || theRole1.multiplicity == xyMANY)
        {
%>
/* ============================================================
 *   Index: Rel_<%=theRole.roleName%><%=Counter%><%Counter++%>_FK1
 * ============================================================
 */
create index Rel_<%=theRole.roleName%><%=Counter%><%Counter++%>_FK1 on
<%=theMClass.name%> (<%=theRole2.roleName%>Id)
go
<%
        }//end if(theRole1.multiplicity == xyZERO_OR_MORE...)
        if(theRole2.multiplicity == xyZERO_OR_MORE
           || theRole2.multiplicity == xyONE_OR_MORE
           || theRole2.multiplicity == xyMANY)
        {
%>
/* ============================================================
 *   Index: Rel_<%=theRole.roleName%><%=Counter%><%Counter++%>_FK2
 * ============================================================
 */
create index Rel_<%=theRole.roleName%><%=Counter%><%Counter++%>_FK2 on
<%=theMClass.name%> (<%=theRole1.roleName%>Id)
go
<%
        }//end if(theRole2.multiplicity == xyZERO_OR_MORE...)
    }// end if(theMClass.hasLinkAssociation())
} // end for (i = 0; i < theMClasses.Count; i++)
%>
<% // for each relation in component
for(i = 0; i < mcomponent.getAssociations().Count; i++) {
    var theAssoc = mcomponent.getAssociations().Item(i);
    var theRole1 = theAssoc.getRoles().Item(0);
    var theRole2 = theAssoc.getRoles().Item(1);
    // if the association has linkclass
```

```
        //  it already handle in the process of
        //  creating link class table
        if(theAssoc.hasLinkClass()) {
            continue;
        }
        if(theRole1.getMetaClass().persistence != xyPERSISTENT
           || theRole2.getMetaClass().persistence != xyPERSISTENT) {
            continue;
        }
        // if the assoc is many to many relation
        if((theRole1.multiplicity == xyMANY
            || theRole1.multiplicity == xyZERO_OR_MORE
            || theRole1.multiplicity == xyONE_OR_MORE)
           &&(theRole2.multiplicity == xyMANY
               || theRole2.multiplicity == xyZERO_OR_MORE
               || theRole2.multiplicity == xyONE_OR_MORE)) {
%>
/* ============================================================
 *   Table: <%=theAssoc.name%>
 * ============================================================
 */
create table <%=theAssoc.name%>
(
    <%=theRole1.roleName%>Id    varchar(40)    not null,
    <%=theRole2.roleName%>Id    varchar(40)    not null,
    constraint PK_<%=theAssoc.name%> primary key (<%=theRole1.roleName%>Id,
<%=theRole2.roleName%>Id)
)
go

/* ============================================================
 *   Index: Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>_FK
 * ============================================================
 */
create index Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>_FK on
<%=theAssoc.name%> (<%=theRole1.roleName%>Id)
go

/* ============================================================
 *   Index: Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>_FK
 * ============================================================
 */
create index Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>_FK on
<%=theAssoc.name%> (<%=theRole2.roleName%>Id)
go
<%
      } // end if((theRole1.multiplicity == xyMANY ...))
   } // end for(i = 0; i < mcomponent.getAssociations().Count; i++)
%>
<% // for each generalization in the component
   //   add referencial integrity constraints
   var theGens = mcomponent.getGeneralizations();
   for(i = 0; i < theGens.Count; i++) {
       var theGen = theGens.Item(i);
       var superClass = theGen.getSuperClass();
       var subClass = theGen.getSubClass();
```

```
        if(superClass.persistence != xyPERSISTENT || subClass.persistence !=
xyPERSISTENT) {
            continue;
        }
%>
/* ============================================================
 *   Reference: Inhr_<%=subClass.name%><%=Counter%><%Counter++%>
 * ============================================================
 */
alter table <%=subClass.name%>
    add constraint Inhr_<%=subClass.name%><%=Counter%><%Counter++%> foreign key
(<%=subClass.name%>Id)
    references <%=superClass.name%> (<%=superClass.name%>Id)
go
<%
   } // end for(i = 0; i < theGens.Count; i++)
%>
<%
   // for each link association in the component
   //   add referencial integrity constraints
   var theAssocs = mcomponent.getAssociations();
   for(i = 0; i < theAssocs.Count; i++) {
       var theAssoc = theAssocs.Item(i);
       // only handle the link association
       if(!theAssoc.hasLinkClass()){
           continue;
       }
       var theRole1 = theAssoc.getRoles().Item(0);
       var theRole2 = theAssoc.getRoles().Item(1);
       if(theRole1.multiplicity == xyEXACTLY_ONE
          || theRole1.multiplicity == xyOPTIONAL) {
%>
/* ============================================================
 *   Reference: Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>
 * ============================================================
 */
alter table <%=theRole2.getMetaClass().name%>
    add constraint Rel_<%=theAssoc.name%><%=Counter%><%Counter++%> foreign key
(<%=theAssoc.getLinkClass().name%>Id)
    references
<%=theAssoc.getLinkClass().name%>(<%=theAssoc.getLinkClass().name%>Id)
go
<%
       }//end if(theRole1.multiplicity == xyEXACTLY_ONE...)
       if(theRole2.multiplicity == xyEXACTLY_ONE
          || theRole2.multiplicity == xyOPTIONAL) {
%>
/* ============================================================
 *   Reference: Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>
 * ============================================================
 */
alter table <%=theRole1.getMetaClass().name%>
    add constraint Rel_<%=theAssoc.name%><%=Counter%><%Counter++%> foreign key
(<%=theAssoc.getLinkClass().name%>Id)
    references
<%=theAssoc.getLinkClass().name%>(<%=theAssoc.getLinkClass().name%>Id)
go
```

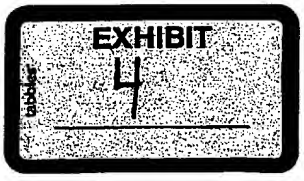
```
<%
       }//end if(theRole2.multiplicity == xyEXACTLY_ONE...)
       if(theRole1.multiplicity == xyZERO_OR_MORE
          || theRole1.multiplicity == xyONE_OR_MORE
          || theRole1.multiplicity == xyMANY)
       {
%>
/* ============================================================
 *   Reference: Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>
 * ============================================================
 */
alter table <%=theAssoc.getLinkClass().name%>
    add constraint Rel_<%=theAssoc.name%><%=Counter%><%Counter++%> foreign key
(<%=theRole2.roleName%>Id)
    references
<%=theRole2.getMetaClass().name%>(<%=theRole2.getMetaClass().name%>Id)
go
<%
       }//end if(theRole1.multiplicity == xyZERO_OR_MORE...)
       if(theRole2.multiplicity == xyZERO_OR_MORE
          || theRole2.multiplicity == xyONE_OR_MORE
          || theRole2.multiplicity == xyMANY)
       {
%>
/* ============================================================
 *   Reference: Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>
 * ============================================================
 */
alter table <%=theAssoc.getLinkClass().name%>
    add constraint Rel_<%=theAssoc.name%><%=Counter%><%Counter++%> foreign key
(<%=theRole1.roleName%>Id)
    references
<%=theRole1.getMetaClass().name%>(<%=theRole1.getMetaClass().name%>Id)
go
<%
       } //end if(theRole2.multiplicity == xyZERO_OR_MORE...)
%>
<%
   } // end for(i = 0; i < theAssocs.Count; i++)
%>
<%
   // for each non-link association in the component
   //   add referencial integrity constraints
   var theAssocs = mcomponent.getAssociations();
   for(i = 0; i < theAssocs.Count; i++) {
       var theAssoc = theAssocs.Item(i);
       var theRole1 = theAssoc.getRoles().Item(0);
       var theRole2 = theAssoc.getRoles().Item(1);
       // the link association already handled
       if(theAssoc.hasLinkClass()){
           continue;
       }
       if(theRole1.getMetaClass().persistence != xyPERSISTENT
          || theRole2.getMetaClass().persistence != xyPERSISTENT) {
           continue;
       }
       // determine where does foreign key go
```

```
        var hasFKOnRole1 = false;
        var hasFKOnRole2 = false;
        if((theRole1.multiplicity == xyMANY
            || theRole1.multiplicity == xyONE_OR_MORE
            || theRole1.multiplicity == xyZERO_OR_MORE)
           &&
           (theRole2.multiplicity == xyEXACTLY_ONE
            || theRole2.multiplicity == xyOPTIONAL)) {
            // [many] to [0,1 or 1]
            hasFKOnRole1 = true;
        } else if((theRole2.multiplicity == xyMANY
                   || theRole2.multiplicity == xyONE_OR_MORE
                   || theRole2.multiplicity == xyZERO_OR_MORE)
                  &&
                  (theRole1.multiplicity == xyEXACTLY_ONE
                   || theRole1.multiplicity == xyOPTIONAL)) {
            // [0,1 or 1] to [many]
            hasFKOnRole2 = true;
        } else if((theRole2.multiplicity == xyEXACTLY_ONE
                   || theRole2.multiplicity == xyOPTIONAL)
                  &&
                  (theRole1.multiplicity == xyEXACTLY_ONE
                   || theRole1.multiplicity == xyOPTIONAL)) {
            // [0,1 or 1] to [0,1 or 1]
            if(theRole1.isAggregate) {
                hasFKOnRole1 = true;
            } else if(theRole2.isAggregate) {
                hasFKOnRole2 = true;
            } else if(theRole1.multiplicity == xyEXACTLY_ONE
                      && theRole2.multiplicity == xyOPTIONAL) {
                hasFKOnRole1 = true;
            } else if(theRole2.multiplicity == xyEXACTLY_ONE
                      && theRole1.multiplicity == xyOPTIONAL) {
                hasFKOnRole2 = true;
            } else {
                hasFKOnRole1 = true;
            }
        }
        if(hasFKOnRole2) {
%>
/* ============================================================
 *   Reference: Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>
 * ============================================================
 */
alter table <%=theRole2.getMetaClass().name%>
    add constraint Rel_<%=theAssoc.name%><%=Counter%><%Counter++%> foreign key
(<%=theRole1.roleName%>Id)
    references
<%=theRole1.getMetaClass().name%>(<%=theRole1.getMetaClass().name%>Id)
go
<%
        } // end if(hasFKOnRole2)
        else if (hasFKOnRole1) {
%>
/* ============================================================
 *   Reference: Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>
 * ============================================================
```

```
 */
alter table <%=theRole1.getMetaClass().name%>
    add constraint Rel_<%=theAssoc.name%><%=Counter%><%Counter++%> foreign key
(<%=theRole2.roleName%>Id)
    references
<%=theRole2.getMetaClass().name%>(<%=theRole2.getMetaClass().name%>Id)
go
<%
       } // end if (hasFKOnRole1)
       // if [many] to [many]
       if((theRole1.multiplicity == xyMANY
           || theRole1.multiplicity == xyONE_OR_MORE
           || theRole1.multiplicity == xyZERO_OR_MORE)
          &&
          (theRole2.multiplicity == xyMANY
           || theRole2.multiplicity == xyONE_OR_MORE
           || theRole2.multiplicity == xyZERO_OR_MORE)) {
%>
/* ============================================================
 *   Reference: Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>
 * ============================================================
 */
alter table <%=theAssoc.name%>
    add constraint Rel_<%=theAssoc.name%><%=Counter%><%Counter++%> foreign key
(<%=theRole1.roleName%>Id)
    references <%=theRole1.getMetaClass().name%>
(<%=theRole1.getMetaClass().name%>Id)
go

/* ============================================================
 *   Reference: Rel_<%=theAssoc.name%><%=Counter%><%Counter++%>
 * ============================================================
 */
alter table <%=theAssoc.name%>
    add constraint Rel_<%=theAssoc.name%><%=Counter%><%Counter++%> foreign key
(<%=theRole2.roleName%>Id)
    references <%=theRole2.getMetaClass().name%>
(<%=theRole2.getMetaClass().name%>Id)
go
<%
       } // end if((theRole1.multiplicity == xyMANY...))
   } // end for(i = 0; i < theAssocs.Count; i++)
%>
<%
//     function ToUpperCaseName(s) {
//         var s1 = new String("");
//         for(i = 0; i < s.length; i++) {
//             var c = s.charAt(i);
//             if("a" <= c && c <= "z") {
//                 s1 = s1 + c.toUpperCase();
//             } else if("A" <= c && c <= "Z") {
//                 s1 = s1 + "_" + c;
//             }
//         }
//         return (s1);
//     }
%>
```

$/MDE1.x/src/pRobot/templates/sql7x2/SQL7x_sql.cmt

	Wwang	8/08/02	4:25p	Labeled '1.86preview5'
	Wwang	1/28/02	2:56p	Labeled 'v 1.85'
	Wwang	1/28/02	12:45p	Labeled 'pre 1.85 '
	Wwang	1/28/02	12:43p	Labeled 'pre 1.85'
37	Wwang	8/27/01	12:58p	Checked in $/MetaPrograms/windowsDNA/pRobot/metaProgram/schema
	Wwang	8/01/01	4:12p	Labeled 'BranchOut-4.0'
36	Whighsmith	4/16/01	4:49p	Checked in $/MetaPrograms/windowsDNA/pRobot/metaProgram/schema
35	Sbutler	11/01/00	3:07p	Checked in $/MetaPrograms/windowsDNA/pRobot/templates/sql7x2
34	Whighsmith	9/25/00	3:29p	Checked in $/MetaPrograms/windowsDNA/pRobot/templates/sql7x2
33	Sbutler	9/22/00	12:46p	Checked in $/MetaPrograms/windowsDNA/pRobot/templates/sql7x2
32	Wwang	7/31/00	5:28p	Checked in $/MetaPrograms/windowDNA/pRobot/templates/sql7x2
	Wwang	7/31/00	4:14p	Labeled 'Release 1.0 Beta 9'
	Wwang	7/31/00	2:06p	Labeled 'Release 1.0 Pre-Beta 9'
	Wwang	6/27/00	11:42a	Labeled 'Release 1.0 Beta 6'
31	Yliu	5/11/00	10:19a	Checked in $/pRobot/templates/sql7x2
30	Wwang	5/10/00	9:44a	Checked in $/pRobot/templates/sql7x2
29	Yliu	5/02/00	4:57p	Checked in $/pRobot/templates/sql7x2
28	Sbutler	3/30/00	4:36p	Checked in $/pRobot/templates/sql7x2
27	Wwang	3/06/00	2:03p	Checked in $/pRobot/templates/sql7x2
26	Wwang	2/08/00	2:06p	Checked in $/pRobot/templates/sql7x2
25	Wwang	1/27/00	4:17p	Checked in $/pRobot/templates/sql7x2
24	Wwang	1/19/00	2:48p	Checked in $/pRobot/templates/sql7x2
23	Wwang	1/19/00	2:46p	Checked in $/pRobot/templates/sql7x2
22	Wwang	1/10/00	9:25a	Checked in $/pRobot/templates/sql7x2
21	Sbutler	12/20/99	5:35p	Checked in $/pRobot/templates/sql7x2



20	Wwang	12/10/99	2:17p	Checked in $/pRobot/templates/sql7x2
19	Wwang	12/10/99	2:07p	Checked in $/pRobot/templates/sql7x2
18	Sbutler	11/18/99	11:15a	Checked in $/pRobot/templates/sql7x2
17	Sbutler	11/18/99	11:10a	Checked in $/pRobot/templates/sql7x2
16	Sbutler	11/10/99	2:48p	Checked in $/pRobot/templates/sql7x2
15	Sbutler	11/09/99	11:33a	Checked in $/pRobot/templates/sql7x2
14	Wwang	10/25/99	11:45a	Checked in $/pRobot/templates/sql7x2
	Wwang	10/22/99	4:47p	Labeled 'enfoTrust2.0a'
	Wwang	10/22/99	3:26p	Labeled 'enfotrust2.0'
13	Wwang	10/07/99	2:26p	Checked in $/pRobot/templates/sql7x2
12	Wwang	8/25/99	9:08a	Checked in $/pRobot/templates/sql7x
	Wwang	8/08/99	3:44p	Labeled 'EnfoTrust1.0'
	Wwang	8/08/99	1:21p	Labeled 'EnfoTrust1.0'
	Wwang	7/15/99	10:22a	Labeled 'Before preserving load Rose'
	Wwang	6/29/99	12:00p	Labeled '06.29.1999 '
	Wwang	6/20/99	9:34a	Labeled '06.20.1999 9:40am '
	Wwang	6/18/99	1:56p	Labeled 'Before move util package out '
	Wwang	6/06/99	4:26p	Labeled '06.06.1999 04:30pm'
	Wwang	6/03/99	2:09p	Labeled '6/3/1999 14:00'
11	Wwang	5/17/99	5:30p	Checked in $/pRobot/templates/sql7x
10	Wwang	5/17/99	3:39p	Checked in $/pRobot/templates/sql7x
9	Wwang	5/17/99	10:29a	Branched at version 9
	Wwang	4/27/99	5:05p	Labeled 'cg 04.27.1999 5:05PM'
8	Wwang	4/14/99	5:37p	Checked in $/xymplementor/templates/db_sql7
7	Wwang	3/13/99	1:11p	Checked in $/xymplementor/templates/db_sql7
6	Wwang	3/04/99	9:52p	Checked in $/xymplementor/templates/db_sql7
5	Acai	1/19/99	4:47p	Checked in $/Xymplementor/templates/db_sql7
4	Acai	1/18/99	5:29p	Checked in $/Xymplementor/templates/db_sql7
3	Acai	1/18/99	10:48a	Checked in $/Xymplementor/templates/db_sql7
2	Acai	1/14/99	11:00p	Checked in $/Xymplementor/templates/db_sql7
1	Acai	1/13/99	3:19p	Created sql7.sql

Package...

Revision : 21
Author : Wei Wang
Date : 10/18/1998

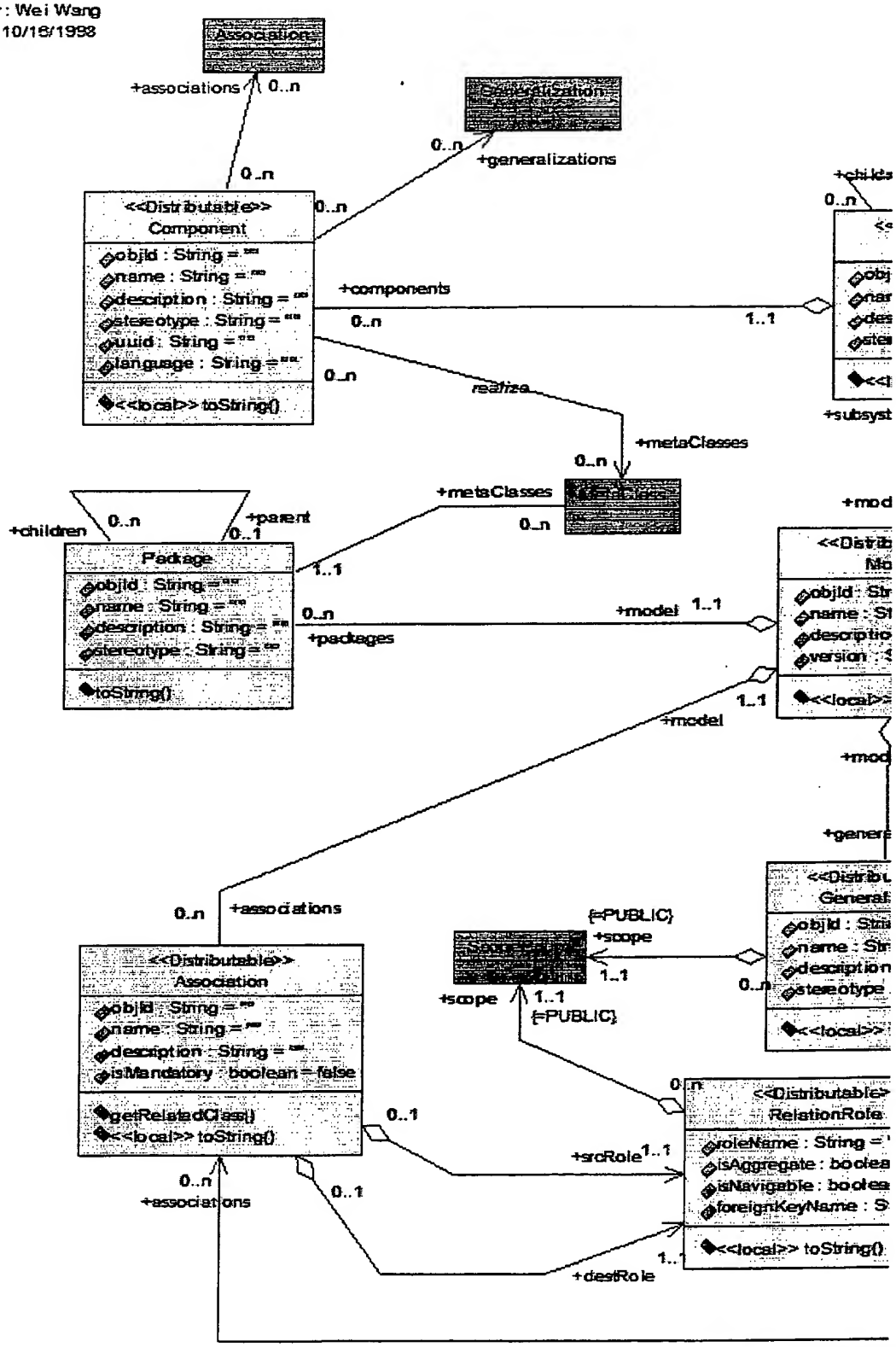
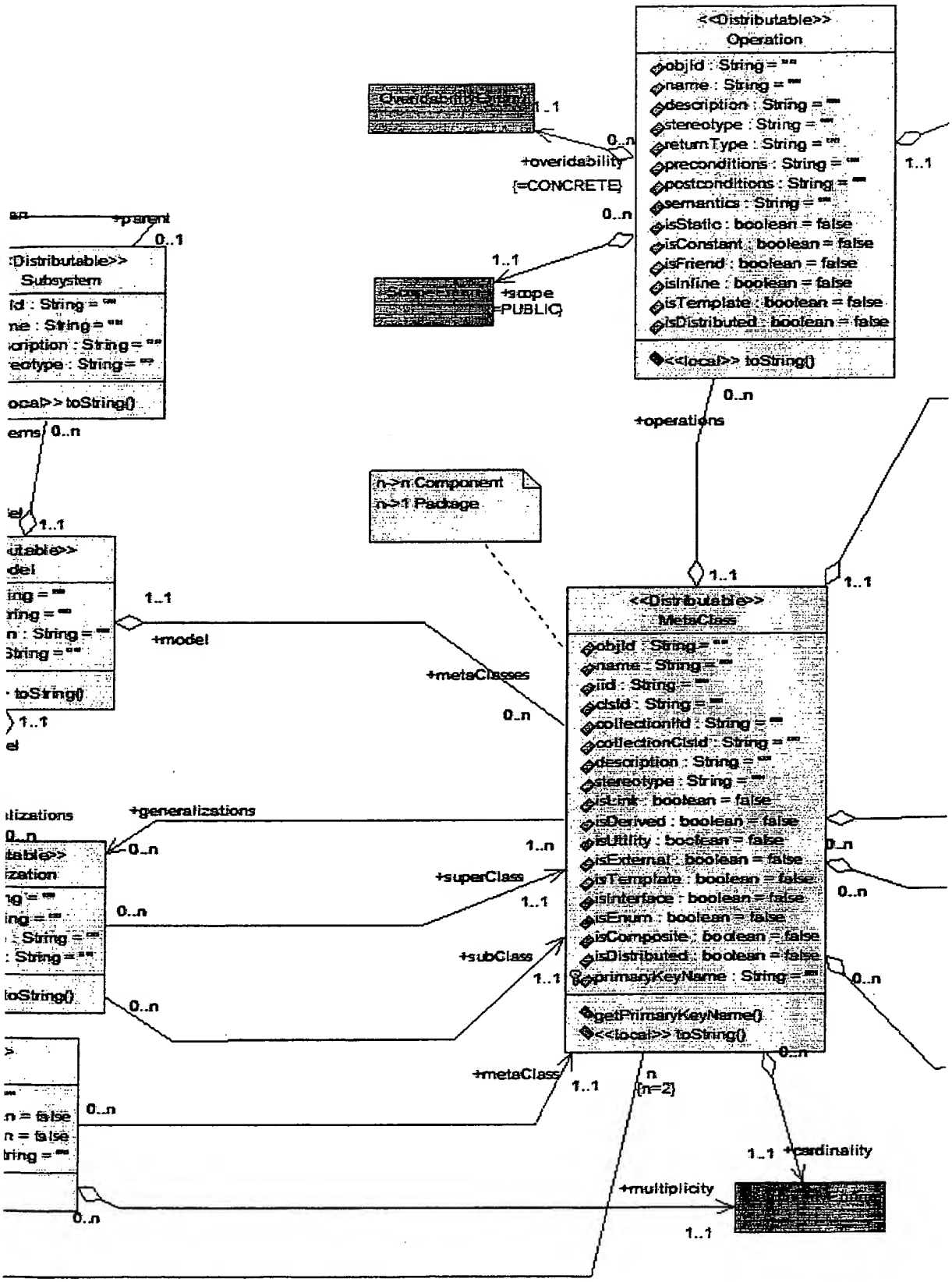
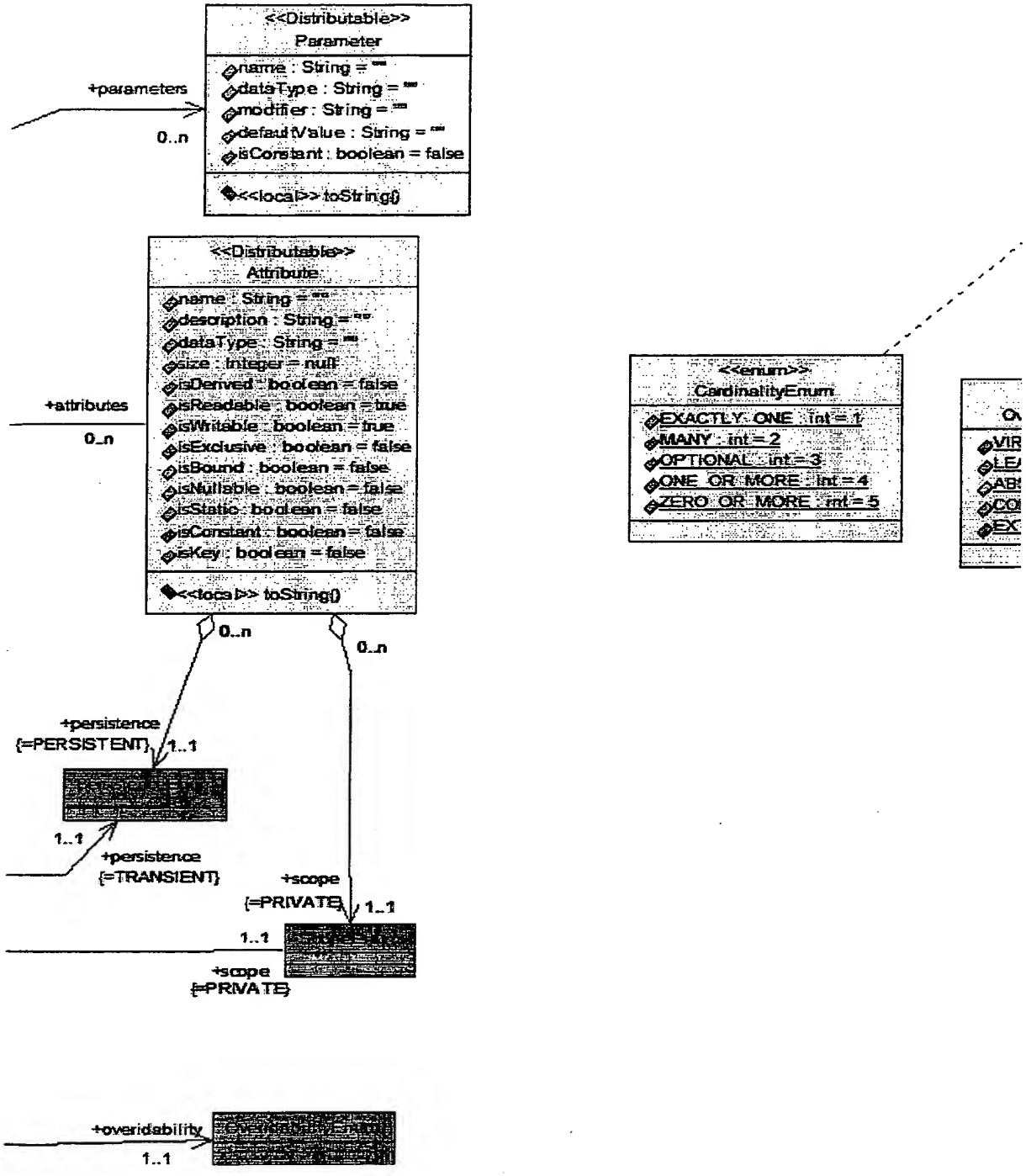
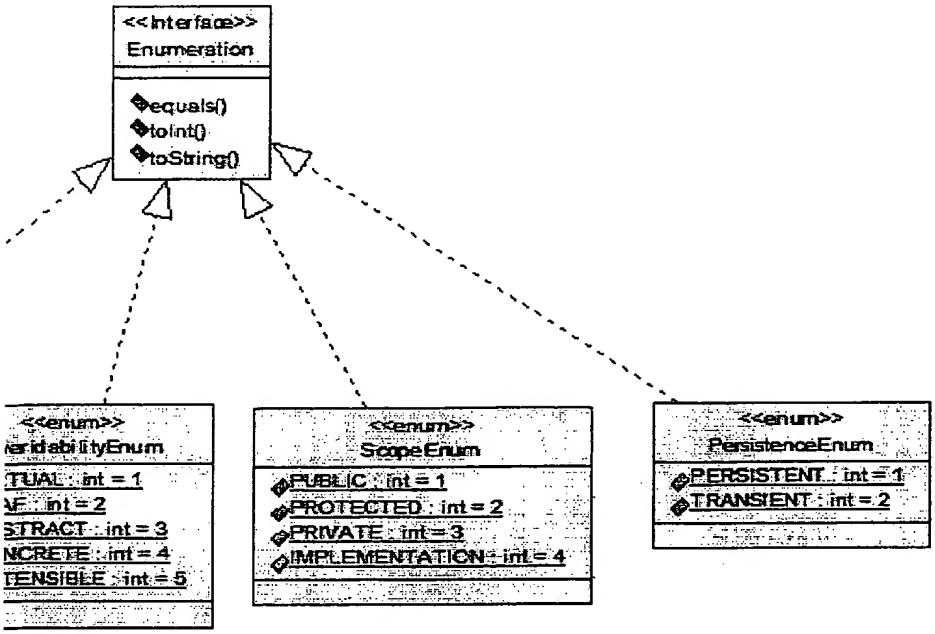


EXHIBIT
5


<<Distributable>>
Operation
objId : String = ""
name : String = ""
description : String = ""
stereotype : String = ""
returnType : String = ""
preconditions : String = ""
postconditions : String = ""
semantics : String = ""
isStatic : boolean = false
isConstant : boolean = false
isFriend : boolean = false
isInline : boolean = false
isTemplate : boolean = false
isDistributed : boolean = false
<<local>> toString()
+overridability
{=CONCRETE}
+scope
=PUBLIC}
+operations
0..n
1..1
+parent
0..1
Subsystem
toString()
n->n Component
n->1 Package
<<Distributable>>
MetaClass
objId : String = ""
name : String = ""
iid : String = ""
clsId : String = ""
collectionId : String = ""
collectionClsId : String = ""
description : String = ""
stereotype : String = ""
isLink : boolean = false
isDerived : boolean = false
isUtility : boolean = false
isExternal : boolean = false
isTemplate : boolean = false
isInterface : boolean = false
isEnum : boolean = false
isComposite : boolean = false
isDistributed : boolean = false
primaryKeyName : String = ""
getPrimaryKeyName()
<<local>> toString()
+model
+metaClasses
+generalizations
+superClass
+subClass
+metaClass
+cardinality
+multiplicity
n
{n=2}


<<Distributable>>
Parameter
name : String = ""
dataType : String = ""
modifier : String = ""
defaultValue : String = ""
isConstant : boolean = false
<<local>> toString()
+parameters
0..n
<<Distributable>>
Attribute
name : String = ""
description : String = ""
dataType : String = ""
size : Integer = null
isDerived : boolean = false
isReadable : boolean = true
isWritable : boolean = true
isExclusive : boolean = false
isBound : boolean = false
isNullable : boolean = false
isStatic : boolean = false
isConstant : boolean = false
isKey : boolean = false
<<local>> toString()
+attributes
0..n
<<enum>>
CardinalityEnum
EXACTLY_ONE : int = 1
MANY : int = 2
OPTIONAL : int = 3
ONE_OR_MORE : int = 4
ZERO_OR_MORE : int = 5
0..n
0..n
+persistence
{=PERSISTENT}
1..1
1..1
+persistence
{=TRANSIENT}
+scope
{=PRIVATE}
1..1
1..1
+scope
{=PRIVATE}
+overridability
1..1


<<Interface>>
Enumeration
equals()
toInt()
toString()
<<enum>>
ScopeEnum
PUBLIC : int = 1
PROTECTED : int = 2
PRIVATE : int = 3
IMPLEMENTATION : int = 4
<<enum>>
PersistenceEnum
PERSISTENT : int = 1
TRANSIENT : int = 2

Package...

Home

Class Diagram: generator / generator

6

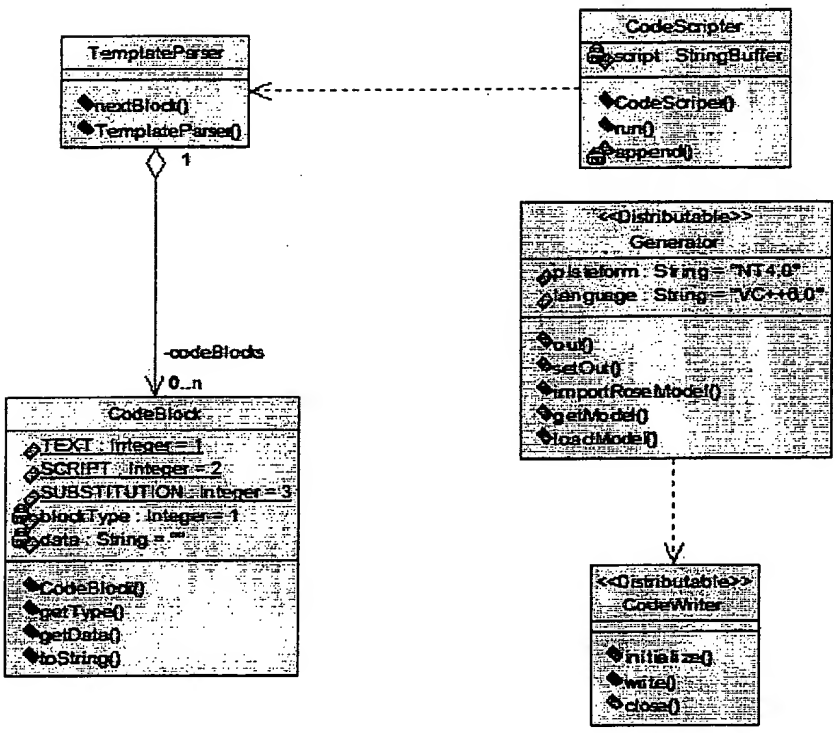


EXHIBIT 6

Home

Use Case Diagram: Use Case View / Main

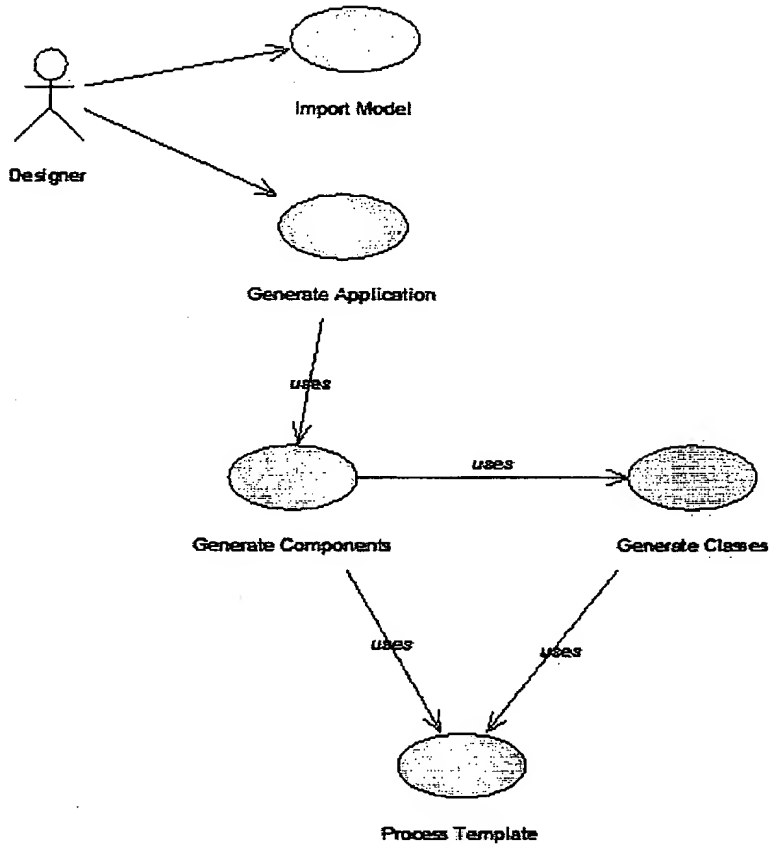


EXHIBIT
7

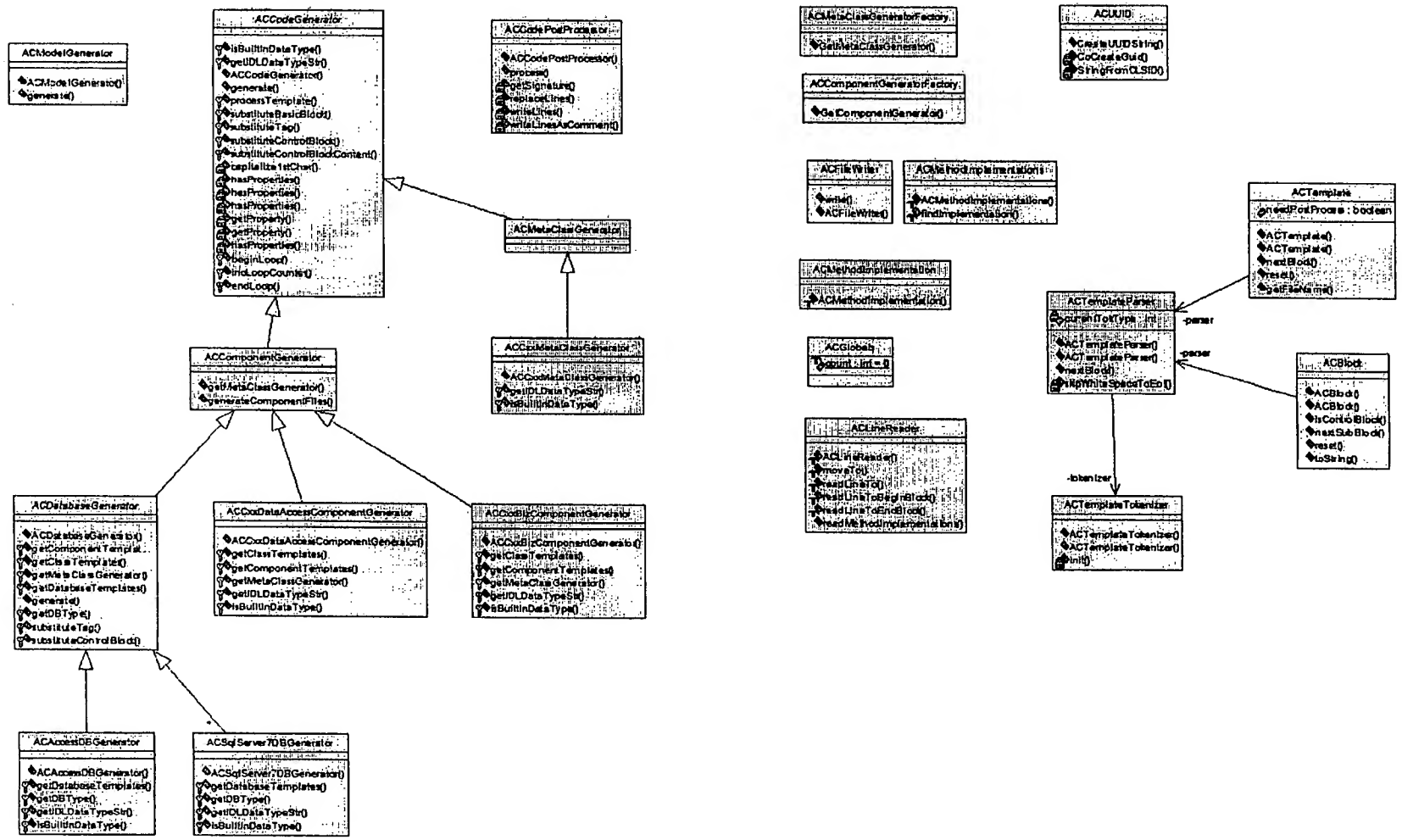

ACModelGenerator
ACCodeGenerator
ACCodePostProcessor
ACMetaClassGeneratorFactory
ACUUID
ACComponentGeneratorFactory
ACFileWriter
ACMethodImplementation
ACTemplate
ACMethodImplementation
ACGlobals
ACTemplateParser
ACBlock
ACLineReader
ACTemplateTokenizer
ACComponentGenerator
ACCodeMetaClassGenerator
ACDatabaseGenerator
ACCxxDataAccessComponentGenerator
ACCodeBizComponentGenerator
ACAccessDBGenerator
ACSqlServer70BGenerator

EXHIBIT
8